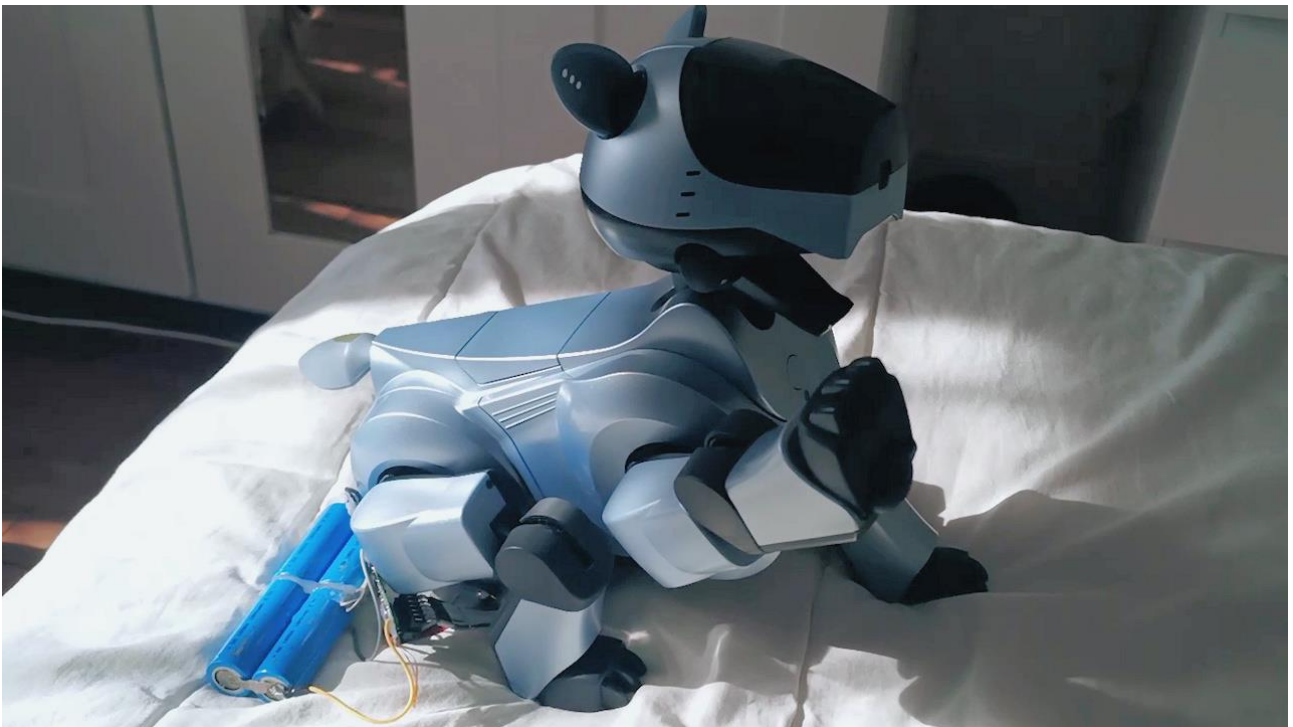


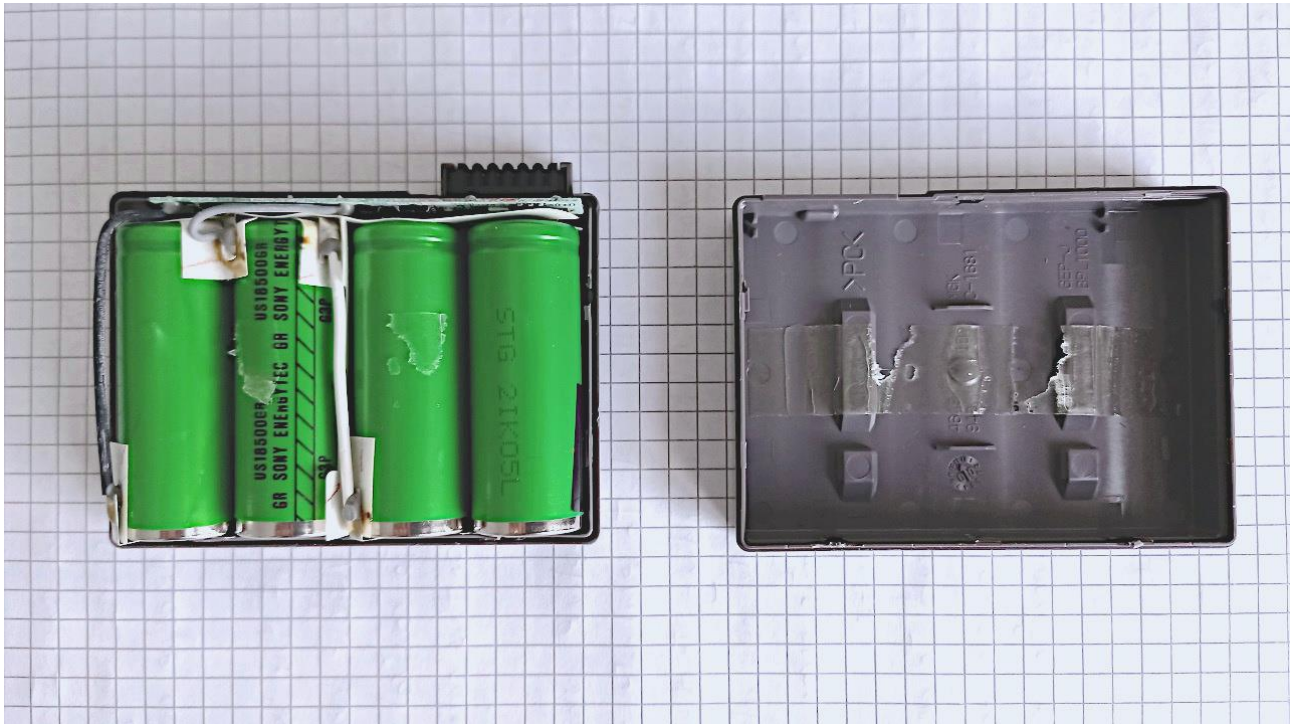
ERA-201B1 re-cell tutorial

How to repair a Sony Aibo ERS-2xx robot battery pack



Abstract

This tutorial is made for happy owners of Sony Aibo robots, more particularly models from the years 2000-2003 including ERS-210, ERS-210A and ERS-220, who would like to repair a used ERA-201B1 battery pack themselves.



Disclaimer

This document and its contents are intended to people enough qualified and experienced in electronics. The information it contains is the result of trial and error which has led to an efficient method of renewing some of the Aibo robot's batteries. This is one way to achieve it and there are certainly others that can work as well.

The procedure is quite technical even for initiated. If you know that you are not yet experienced or you do not have the necessary equipment, it is not recommended that you undertake this operation yourself, as you may permanently damage your battery or even suffer a fire in case of Li-Ion cell short circuit.

[BattMon 2.0](#) repository, which include this tutorial, is provided on Github under MIT license. Its author cannot be held responsible for the consequences that you will make of its use. Be careful, read this document in its entirety and take precautions before you start.

Version history

Revision	Date	Changes	Author
1.0	2022-02-27	First release	Loïc POLLIER

Table of contents

Abstract	2
Disclaimer	2
Version history	2
1. Gather tools and equipment	4
2. Open the battery case and remove the old cells	5
3. Weld the new cells and reassemble them in pack	6
4. Reset the EEPROM parameters	7
5. Solder the pack to the board and reinitialize the battery protection	13
6. Reintegrate all in the case	14
Tips and tricks	15
Appendix A.....	16
Appendix B.....	16

1. Gather tools and equipment

In addition to a solid background in electronics, you need tools and equipment to perform the complete battery replacement operation. Below is the minimal set of material.

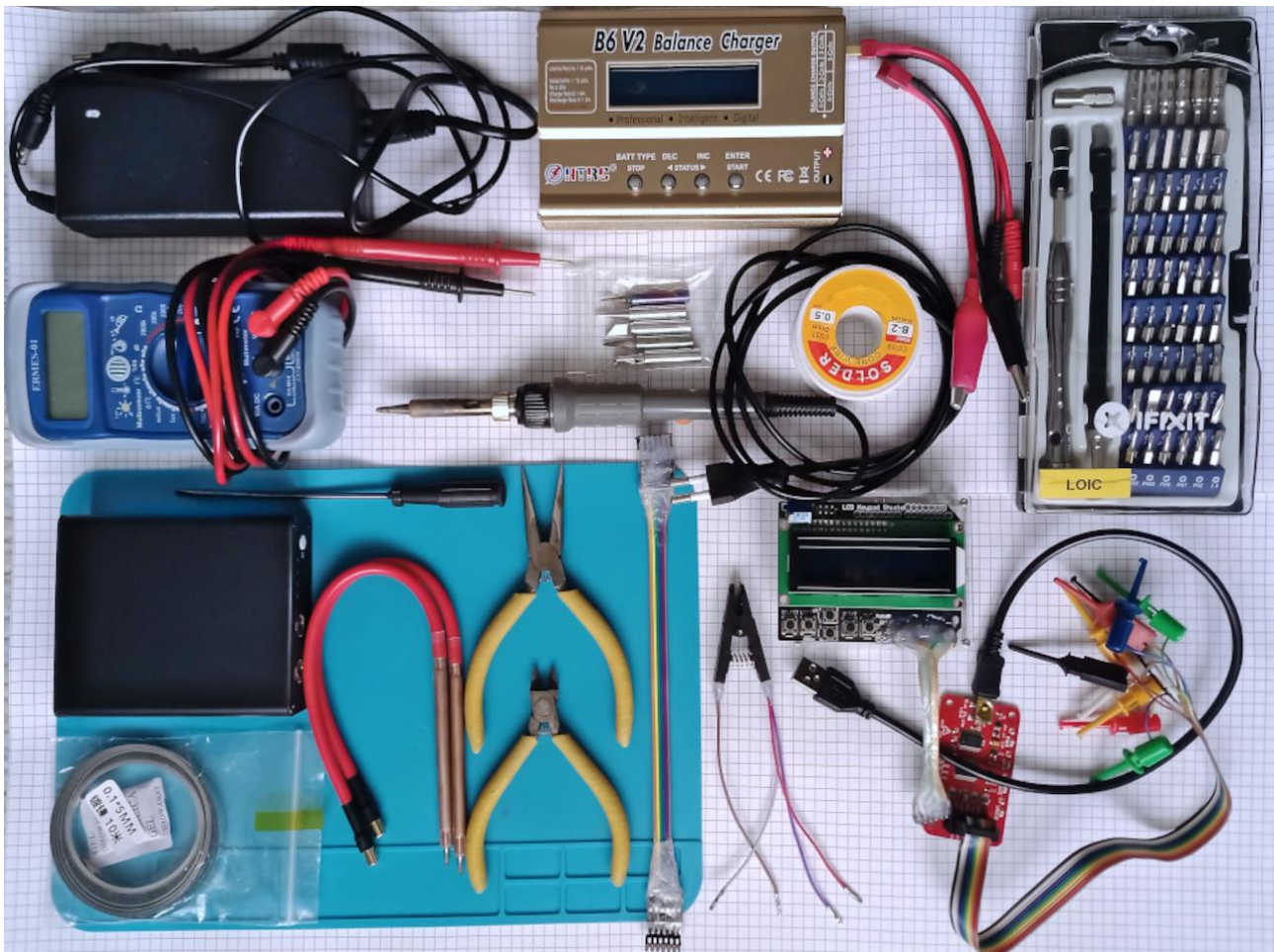
Minimum set:

- Electronic tools (flat screwdriver, flat pliers, cutting pliers, multimeter, electric wires, insulating tape, soldering iron, tin coil, spot welding machine, nickel roller).
- 1 I2C EEPROM memory programmer (e.g. Bus Pirate) connected to a computer.
- 4 new 18500 Lithium-Ion cells.

Moreover, this recommended set, in addition to the minimum set, will certainly guarantee a higher success rate by measuring the actual capacity of the battery and checking its characteristics before, during and after the repair.

Recommended set:

- 1 SMBus battery monitoring tool kit (e.g. BattMon 2.0).
- 1 Li-Ion battery charger (e.g. HTRC B6).
- 1 SO-8 IC test clamp.



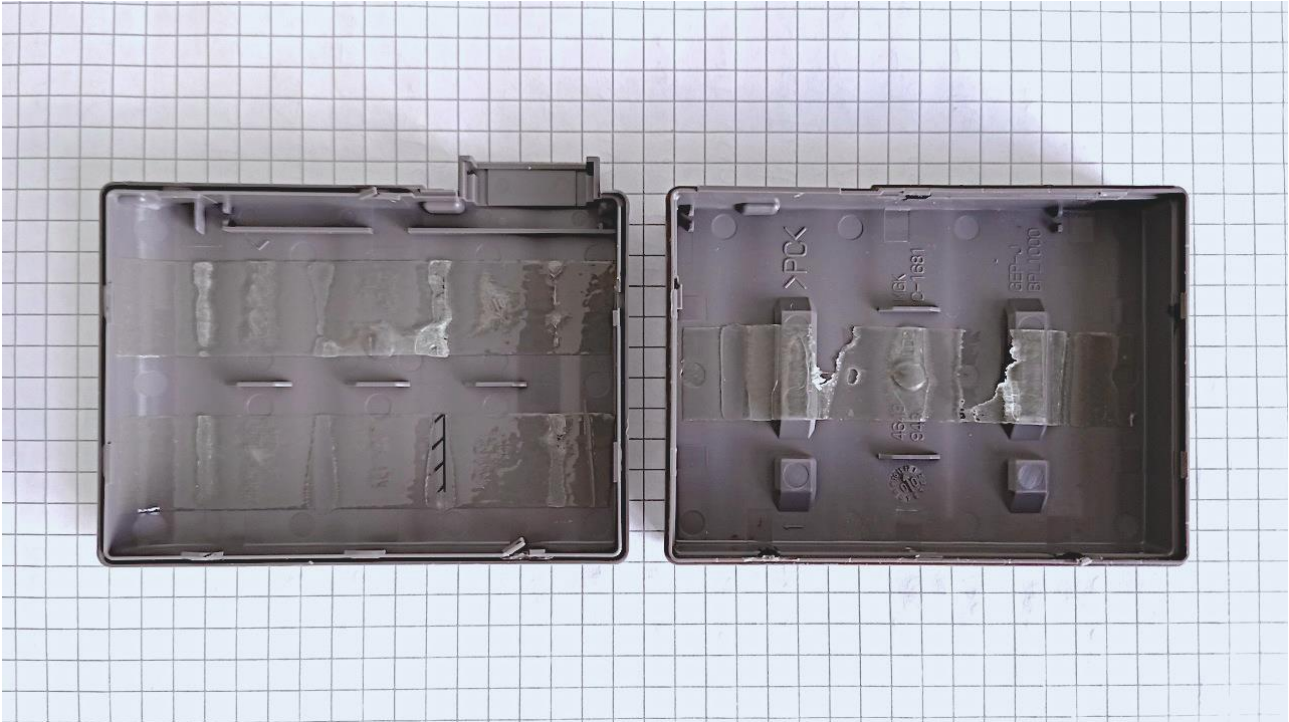
You can find links in Appendices A and B on how to obtain the references for the equipment, the documentation for the tools and the datasheet for the main components.

To summarize, the repair can be broken down into three stages: Replace the cells; Reset some of the EEPROM parameters; Reinitialize the first level battery protection.

2. Open the battery case and remove the old cells

Now that you are equipped, we can start by opening the case to extract the battery pack and remove the used cells. It is recommended that the pack is completely discharged.

This step can be done with a fairly wide and thin flat screwdriver. But to avoid damaging the case too much, you can also use a thin and long flat blade by gently levering it along its entire length.

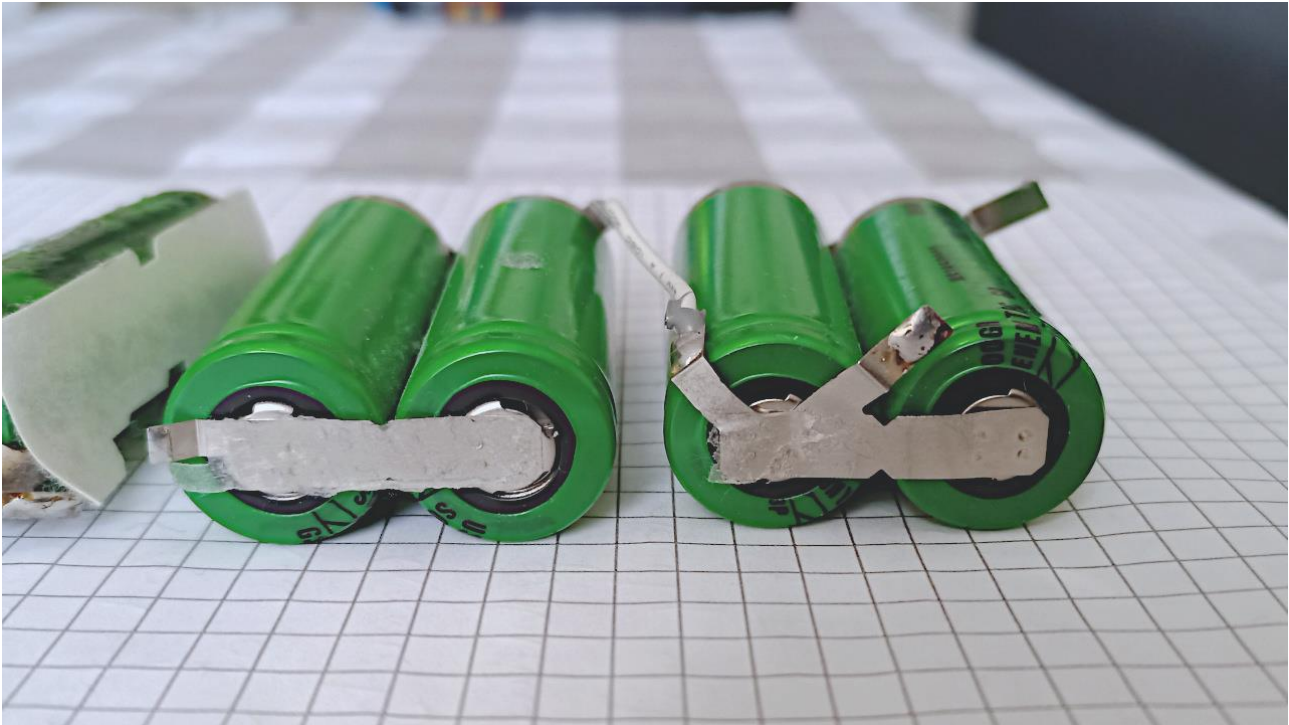


What keeps the two parts of the case around the cells is both strong tape and nine retaining clips on all sides. It is pretty hard not to break a few of them. Take your time.



3. Weld the new cells and reassemble them in pack

After taking the pack out of its case, you can unsolder the nickel plate (BAT+) and the two gray (BATM) and black (BAT-) wires from the PCB. Watch out for short circuits here.

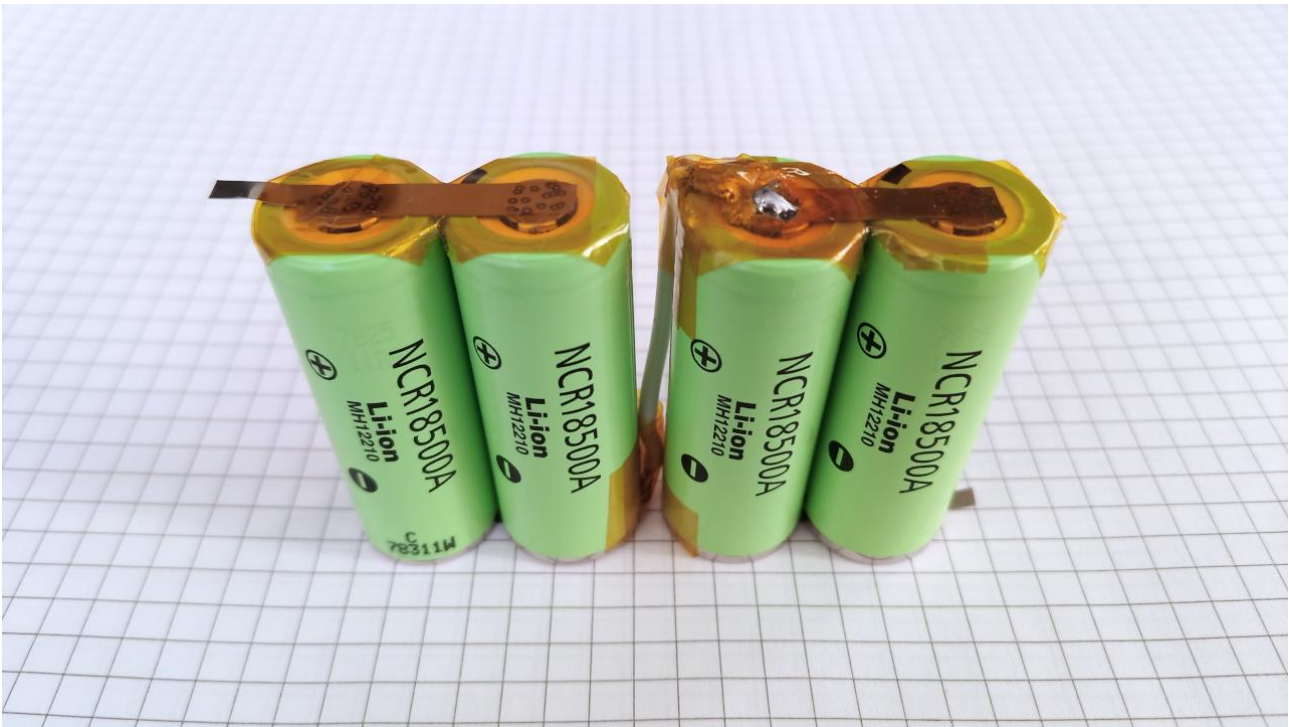


If you have a Lithium Ion battery charger, it is time to individually measure the real capacity of each new cell and thus verify that it is at least as good as the original 1150mAh.

Depending on the battery manufacturer, it often happens that the actual capacity of cheap cells is much lower than that indicated. It is advisable to use brand cells like NCR18500A which have a real capacity close to 2000mAh, as well as a lower internal resistance.



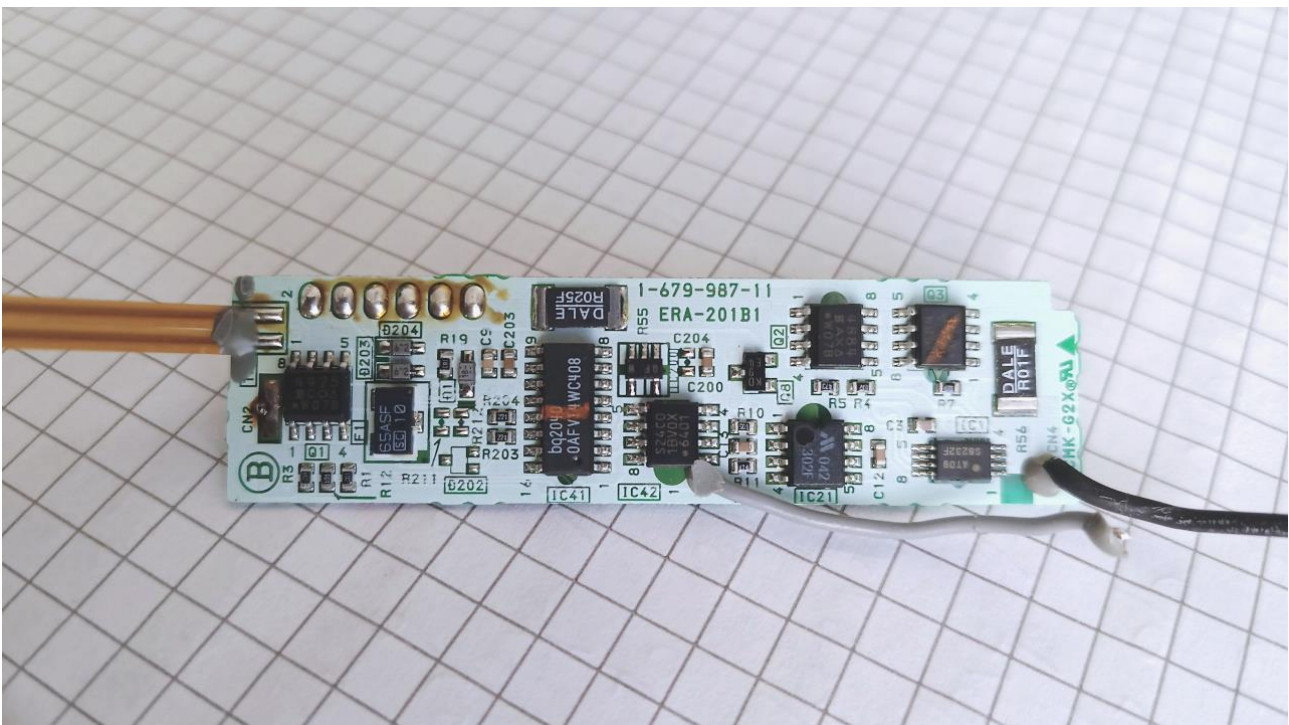
Once the old cells are separated from the PCB, you can continue by reassembling the pack with the new ones. It is better to solder the cells in the same way as the original. For this, you can use a spot welding machine with a nickel roller instead of a soldering iron, in order to reduce the thickness while avoiding damage to them by overheating.



Finally, solder the white wire to serially connect the two sets of cells welded in parallel. It is essential to add a thin layer of electrical insulation such as Kapton tape around the welds.

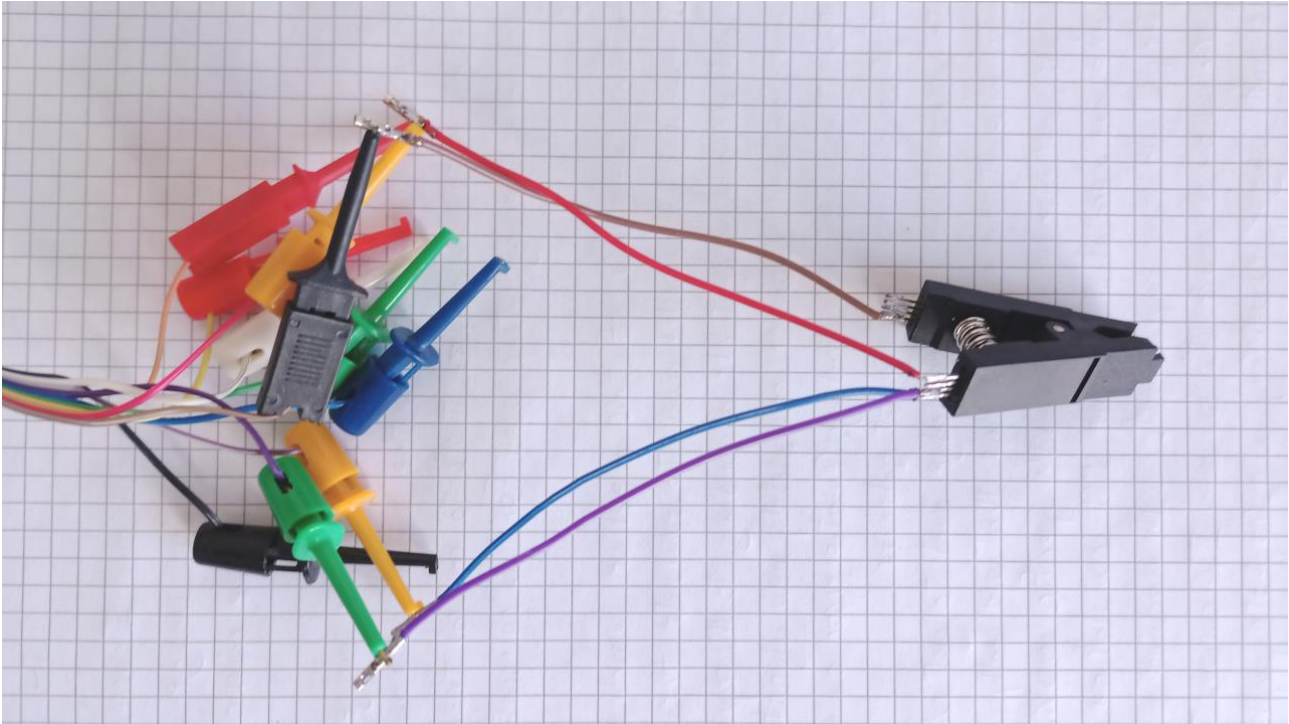
4. Reset the EEPROM parameters

We are now going to update some of the parameters saved in the 24LC01 EEPROM memory IC42 which are used by the bq2040 gas gauge IC41.

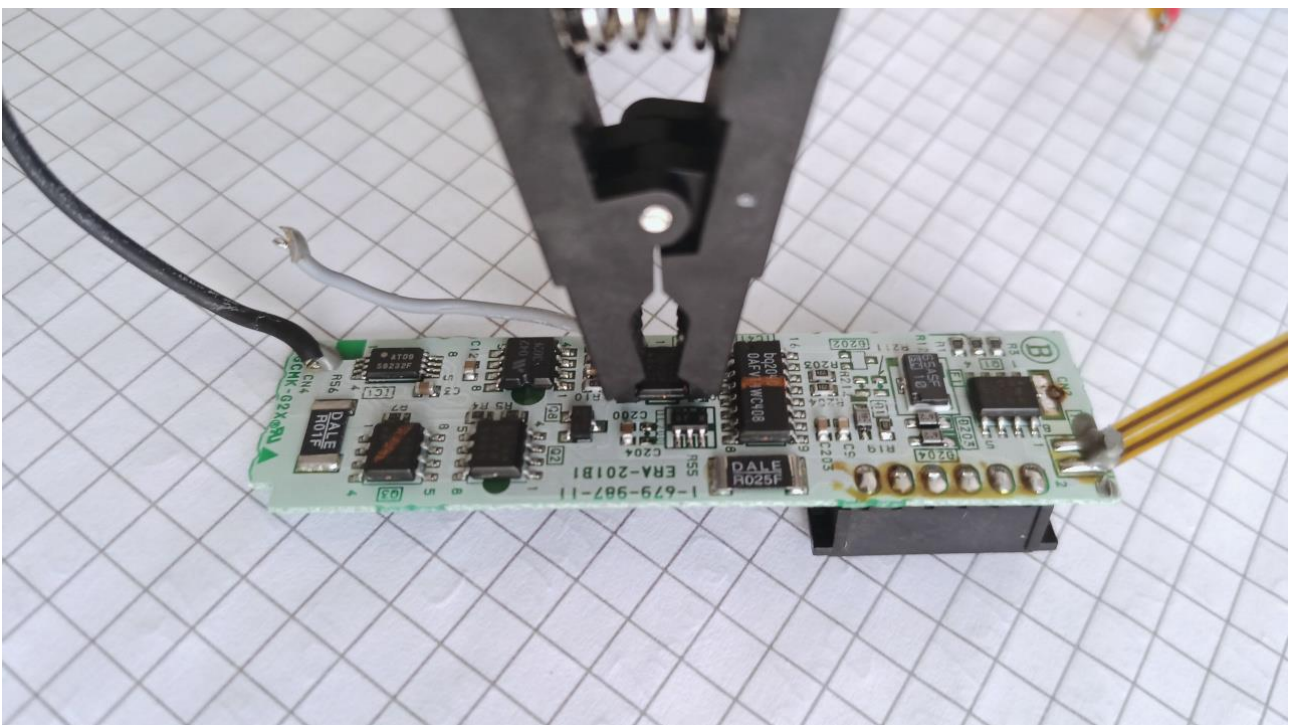


To do this, you will have to use a memory programmer such as the Bus Pirate, interfaced to the EEPROM IC42 via the I2C bus.

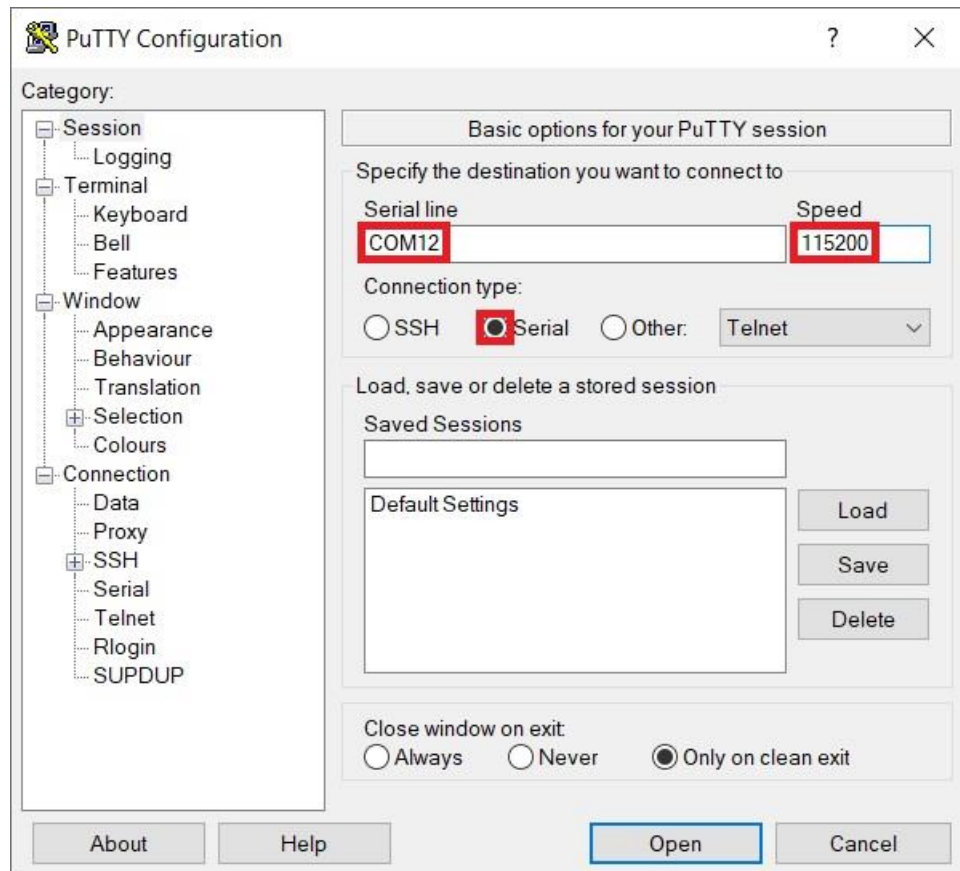
As indicated in the 24LC01 datasheet, four wires must be connected to achieve the physical interface: VCC (+3.3V), VSS (0V), SDA (I2C Data/ Address) and SCL (I2C Clock). You can either solder the necessary wires to TP202, TP209, TP203 and TP204 respectively, or you can create a test clamp in SO-8 format, like the one below.



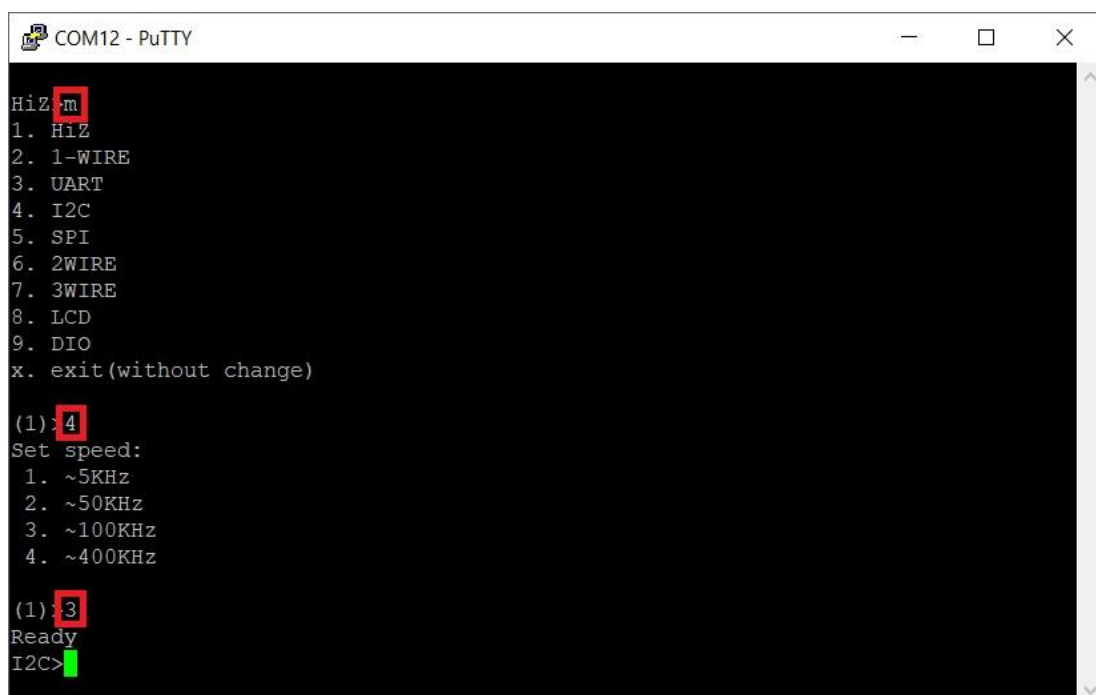
In this case, connect it well so that the pins coincide correctly. If necessary, you can refer to 24LC01 datasheet, where the direction and polarity of the EEPROM is indicated relative to the position of pin 1 represented by a dot or notch.



Afterwards, you can plug the Bus Pirate USB cable to your computer. Here we will use a Windows PC with PuTTY software installed, but it is pretty much the same directly with a Mac or Linux terminal.



This involves connecting to the Bus Pirate by selecting the "serial" option with the default values of 115200 bit/s in "Speed" and your USB COM port number in "Serial line". To find out your port number, go to the Windows menu Device Manager > Ports (COM & LPT), which is in this example COM12.



We will now communicate with the Bus Pirate by command line. Click the "Open" button and start by entering the "m" command to display the main menu as above. You must then select I2C by entering "4" and a standard speed rate of 100KHz by entering "3".

If all went well, you can supply IC42 with the "w" command (or cut off the power with "w"). And "v" allows you to check the voltage which should be close to 3.3V in "2. (RD)" column.

```

(1)>3
Ready
I2C: v
Pinstates:
1. (BR)  2. (RD)  3. (OR)  4. (YW)  5. (GN)  6. (BL)  7. (PU)  8. (GR)  9. (WT)  0. (Blk)
GND      3.3V    5.0V    ADC     VPU     AUX     SCL     SDA     -       -
P        P      P      I      I      I      I      I      I      I
GND      0.00V   0.00V   0.00V   0.00V   L       L       L       L       L
I2C: w
Power supplies ON
I2C: v
Pinstates:
1. (BR)  2. (RD)  3. (OR)  4. (YW)  5. (GN)  6. (BL)  7. (PU)  8. (GR)  9. (WT)  0. (Blk)
GND      3.3V    5.0V    ADC     VPU     AUX     SCL     SDA     -       -
P        P      P      I      I      I      I      I      I      I
GND      3.33V   4.98V   0.00V   0.00V   L       H       H       L       H
I2C> (1)
Searching I2C address space. Found devices at:
0xA0(0x50 W) 0xA1(0x50 R) 0xA2(0x51 W) 0xA3(0x51 R) 0xA4(0x52 W) 0xA5(0x52 R) 0xA6(0x53 W) 0xA7(0x53 R) 0xA8(0x54 W) 0xA9(0x54 R) 0xAA(0x55 W) 0xAB(0x55 R) 0xAC(0x56 W) 0xAD(0x56 R) 0xAE(0x57 W) 0xAF(0x57 R)
I2C>
  
```

Enter the command "(1)" to verify that the EEPROM memory is present on the I2C bus. You should see a response like this: "0xA0(0x50 W) 0xA1(0x50 R)..."

```

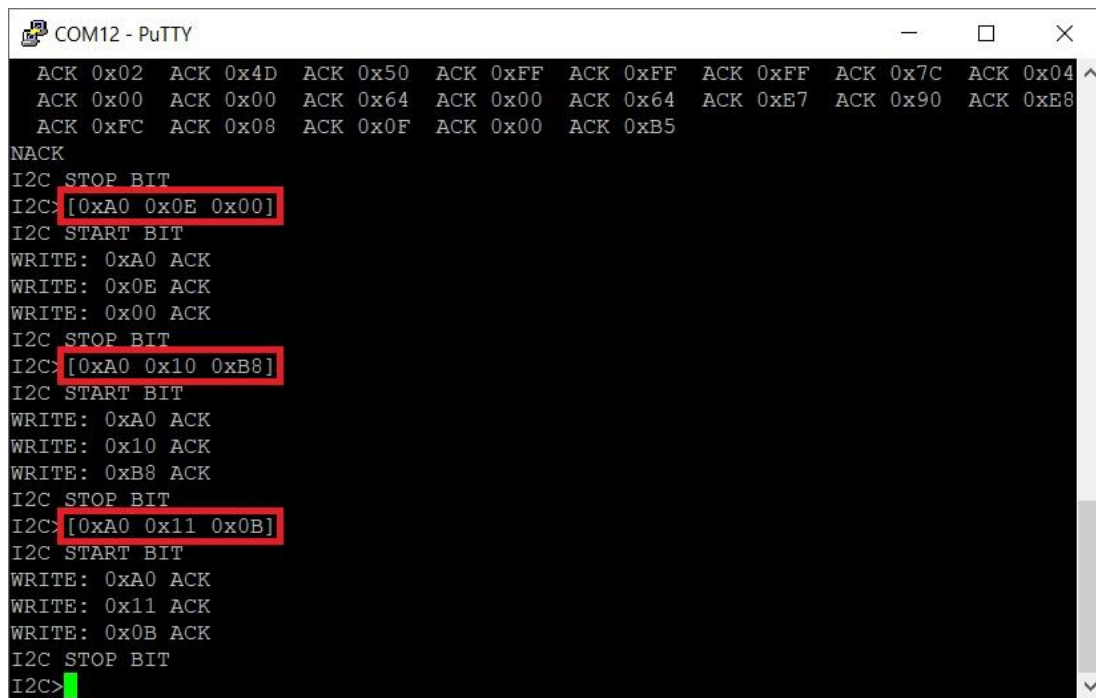
(0x56 W) 0xAD(0x56 R) 0xAE(0x57 W) 0xAF(0x57 R)
I2C> [0xA0 0x00 [0xA1 r:0x65]]
I2C START BIT
WRITE: 0xA0 ACK
WRITE: 0x00 ACK
I2C START BIT
WRITE: 0xA1 ACK
READ: 0x64 ACK 0x5B ACK 0x0A ACK 0x00 ACK 0xE6 ACK 0x00 ACK 0xFF ACK 0xFF
      ACK 0xE8 ACK 0x03 ACK 0xD0 ACK 0x20 ACK 0x80 ACK 0x00 ACK 0x2A ACK 0x00
      ACK 0xFC ACK 0x08 ACK 0xE8 ACK 0x1C ACK 0x10 ACK 0x00 ACK 0x87 ACK 0x29
      ACK 0x28 ACK 0x0B ACK 0xD0 ACK 0x07 ACK 0x00 ACK 0x00 ACK 0x00 ACK 0x00
      ACK 0x0A ACK 0x53 ACK 0x6F ACK 0x6E ACK 0x79 ACK 0x20 ACK 0x43 ACK 0x6F
      ACK 0x72 ACK 0x70 ACK 0x2E ACK 0xFF ACK 0x88 ACK 0x13 ACK 0x08 ACK 0xFF
      ACK 0x07 ACK 0x45 ACK 0x52 ACK 0x41 ACK 0x32 ACK 0x30 ACK 0x31 ACK 0x42
      ACK 0x1A ACK 0xFF ACK 0x9C ACK 0xFF ACK 0x00 ACK 0xB0 ACK 0x00 ACK 0x38
      ACK 0x04 ACK 0x4C ACK 0x49 ACK 0x4F ACK 0x4E ACK 0xFF ACK 0x22 ACK 0x1D
      ACK 0x0B ACK 0x91 ACK 0x5F ACK 0xFF ACK 0x9D ACK 0x96 ACK 0x7C ACK 0x05
      ACK 0x02 ACK 0x4D ACK 0x50 ACK 0xFF ACK 0xFF ACK 0xFF ACK 0x7C ACK 0x04
      ACK 0x00 ACK 0x00 ACK 0x64 ACK 0x00 ACK 0x64 ACK 0xE7 ACK 0x90 ACK 0xE8
      ACK 0xFC ACK 0x08 ACK 0x0F ACK 0x00 ACK 0xB5
NACK
I2C STOP BIT
I2C>
  
```

You can read the memory contents with this command: "[0xA0 0x00 [0xA1 r:0x65]]". Be sure to copy-paste this original data into a separate text file, you may find it useful later.

Among the contents of the memory you can see surrounded in orange three parameters that we are going to modify: Cycle count, Design capacity and Full-charge capacity.

So far we have sent read commands from memory, we will now have to write in it. The values are coded in hexadecimal, so we have to convert from decimal to hexadecimal what we want to write. Most calculators, like the one in Windows, offer this function.

The initial values in this example are: Cycle count: "0x002A" -> 42 full charge cycles, Design capacity: "0x08FC" -> 2300mAh, Full-charge capacity: "0x08FC" -> 2300mAh.

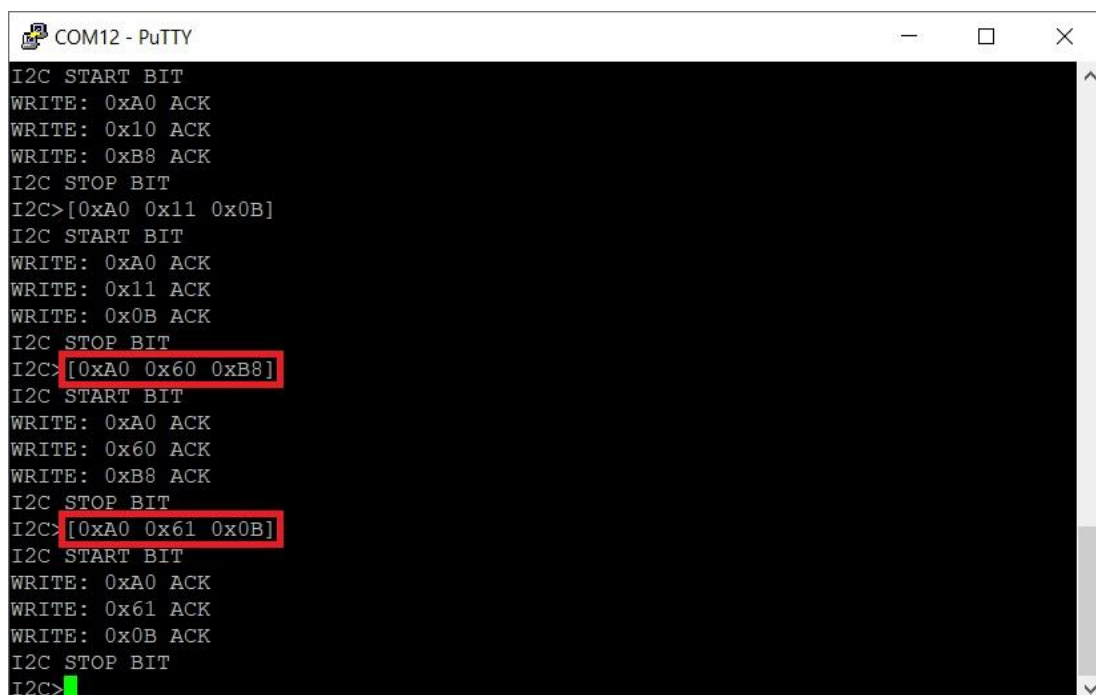


```

COM12 - PuTTY
ACK 0x02 ACK 0x4D ACK 0x50 ACK 0xFF ACK 0xFF ACK 0xFF ACK 0x7C ACK 0x04
ACK 0x00 ACK 0x00 ACK 0x64 ACK 0x00 ACK 0x64 ACK 0xE7 ACK 0x90 ACK 0xE8
ACK 0xFC ACK 0x08 ACK 0x0F ACK 0x00 ACK 0xB5
NACK
I2C STOP BIT
I2C> [0xA0 0x0E 0x00]
I2C START BIT
WRITE: 0xA0 ACK
WRITE: 0x0E ACK
WRITE: 0x00 ACK
I2C STOP BIT
I2C> [0xA0 0x10 0xB8]
I2C START BIT
WRITE: 0xA0 ACK
WRITE: 0x10 ACK
WRITE: 0xB8 ACK
I2C STOP BIT
I2C> [0xA0 0x11 0x0B]
I2C START BIT
WRITE: 0xA0 ACK
WRITE: 0x11 ACK
WRITE: 0x0B ACK
I2C STOP BIT
I2C>

```

Imagine that we individually measured our four cells with an actual capacity of around 1500mAh each. It will therefore be necessary to write in the EEPROM 3000mAh, the double corresponding to the total capacity of the pack, so 0x0BB8 in hexadecimal.



```

COM12 - PuTTY
I2C START BIT
WRITE: 0xA0 ACK
WRITE: 0x10 ACK
WRITE: 0xB8 ACK
I2C STOP BIT
I2C> [0xA0 0x11 0x0B]
I2C START BIT
WRITE: 0xA0 ACK
WRITE: 0x11 ACK
WRITE: 0x0B ACK
I2C STOP BIT
I2C> [0xA0 0x60 0xB8]
I2C START BIT
WRITE: 0xA0 ACK
WRITE: 0x60 ACK
WRITE: 0xB8 ACK
I2C STOP BIT
I2C> [0xA0 0x61 0x0B]
I2C START BIT
WRITE: 0xA0 ACK
WRITE: 0x61 ACK
WRITE: 0x0B ACK
I2C STOP BIT
I2C>

```


In addition, the charge cycle counter must also be reset. For all that, we will enter this series of commands, as seen in red above:

```
"[0xA0 0x0E 0x00]" // Cycle count LSB set to 0x00 at memory address 0x0E
"[0xA0 0x0F 0x00]" // Cycle count MSB set to 0x00 at memory address 0x0F
"[0xA0 0x10 0xB8]" // Design capacity LSB set to 0xB8 at memory address 0x10
"[0xA0 0x11 0x0B]" // Design capacity MSB set to 0x0B at memory address 0x11
"[0xA0 0x60 0xB8]" // Full-charge capacity LSB set to 0xB8 at memory address 0x60
"[0xA0 0x61 0x0B]" // Full-charge capacity MSB set to 0x0B at memory address 0x61
```

We'll adjust the capacity value in memory to the double that of your cells. For example, if you have NCR18500A cells of around 2000mAh, you should use 4000mAh, so 0x0FA0 in hexadecimal.

Finally, check the new memory contents: "[0xA0 0x00 [0xA1 r:0x65]]". You should see the updated values as in green below.

```
COM12 - PuTTY
WRITE: 0x0B ACK
I2C STOP BIT
I2C> [0xA0 0x00 [0xA1 r:0x65]]
I2C START BIT
WRITE: 0xA0 ACK
WRITE: 0x00 ACK
I2C START BIT
WRITE: 0xA1 ACK
READ: 0x64 ACK 0x5B ACK 0x0A ACK 0x00 ACK 0xE6 ACK 0x00 ACK 0xFF ACK 0xFF
ACK 0xE8 ACK 0x03 ACK 0xD0 ACK 0x20 ACK 0x80 ACK 0x00 ACK 0x00 ACK 0x00
ACK 0xB8 ACK 0x0B ACK 0xE8 ACK 0x1C ACK 0x10 ACK 0x00 ACK 0x87 ACK 0x29
ACK 0x28 ACK 0x0B ACK 0xD0 ACK 0x07 ACK 0x00 ACK 0x00 ACK 0x00 ACK 0x00
ACK 0x0A ACK 0x53 ACK 0x6F ACK 0x6E ACK 0x79 ACK 0x20 ACK 0x43 ACK 0x6F
ACK 0x72 ACK 0x70 ACK 0x2E ACK 0xFF ACK 0x88 ACK 0x13 ACK 0x08 ACK 0xFF
ACK 0x07 ACK 0x45 ACK 0x52 ACK 0x41 ACK 0x32 ACK 0x30 ACK 0x31 ACK 0x42
ACK 0x1A ACK 0xFF ACK 0x9C ACK 0xFF ACK 0x00 ACK 0xB0 ACK 0x00 ACK 0x38
ACK 0x04 ACK 0x4C ACK 0x49 ACK 0x4F ACK 0x4E ACK 0xFF ACK 0x22 ACK 0x1D
ACK 0x0B ACK 0x91 ACK 0x5F ACK 0xFF ACK 0x9D ACK 0x96 ACK 0x7C ACK 0x05
ACK 0x02 ACK 0x4D ACK 0x50 ACK 0xFF ACK 0xFF ACK 0xFF ACK 0x7C ACK 0x04
ACK 0x00 ACK 0x00 ACK 0x64 ACK 0x00 ACK 0x64 ACK 0xE7 ACK 0x90 ACK 0xE8
ACK 0xB8 ACK 0x0B ACK 0x0F ACK 0x00 ACK 0xB5
NACK
I2C STOP BIT
I2C>
```

In the end, you can turn off the power with "w" and unplug everything. There are other settings in memory, but it is safe not to change them.

But if you find that you made a mistake, you can always rewrite the contents of your EEPROM with the original data you saved in a text file.

Each command starting with "0xA0" writes a value in hexadecimal to an address in hexadecimal in this way: "[0xA0 address value]"

Here is an example to write 0xAB in a memory block at address 0x65 and then read it:

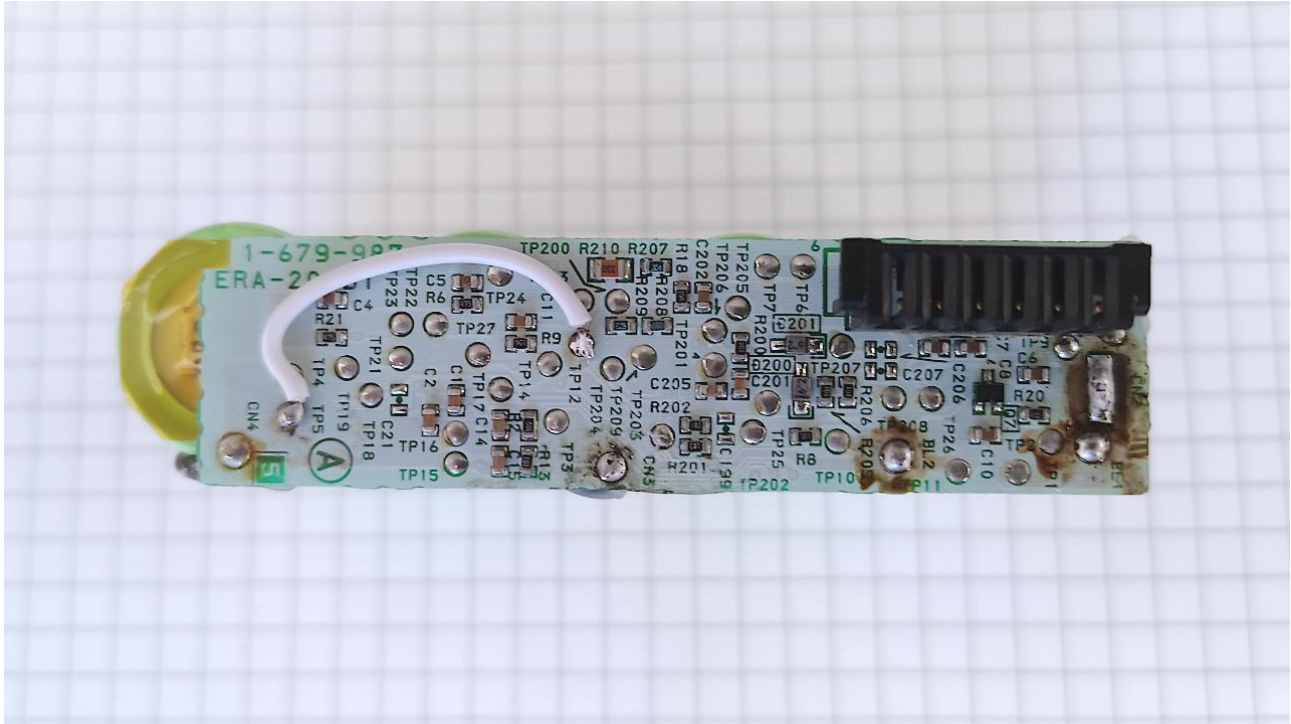
```
"[0xA0 0x65 0xAB]"
"[0xA0 0x65 [0xA1 r]]"
```

And another example to write and read a row of 8 memory blocks starting at address 0x65:

```
"[0xA0 0x65 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07]"
"[0xA0 0x65 [0xA1 r:8]]"
```

5. Solder the pack to the board and reinitialize the battery protection

Before closing the case, the last step is to solder the pack to the PCB and reinitialize the reference of the first level protection circuit IC1 between BAT- and GND/PACK- voltages, which consists of simultaneously short-circuiting the source of transistors Q2 and Q3. However, if you do it directly, you will probably destroy fuse F1 via transistor Q8.

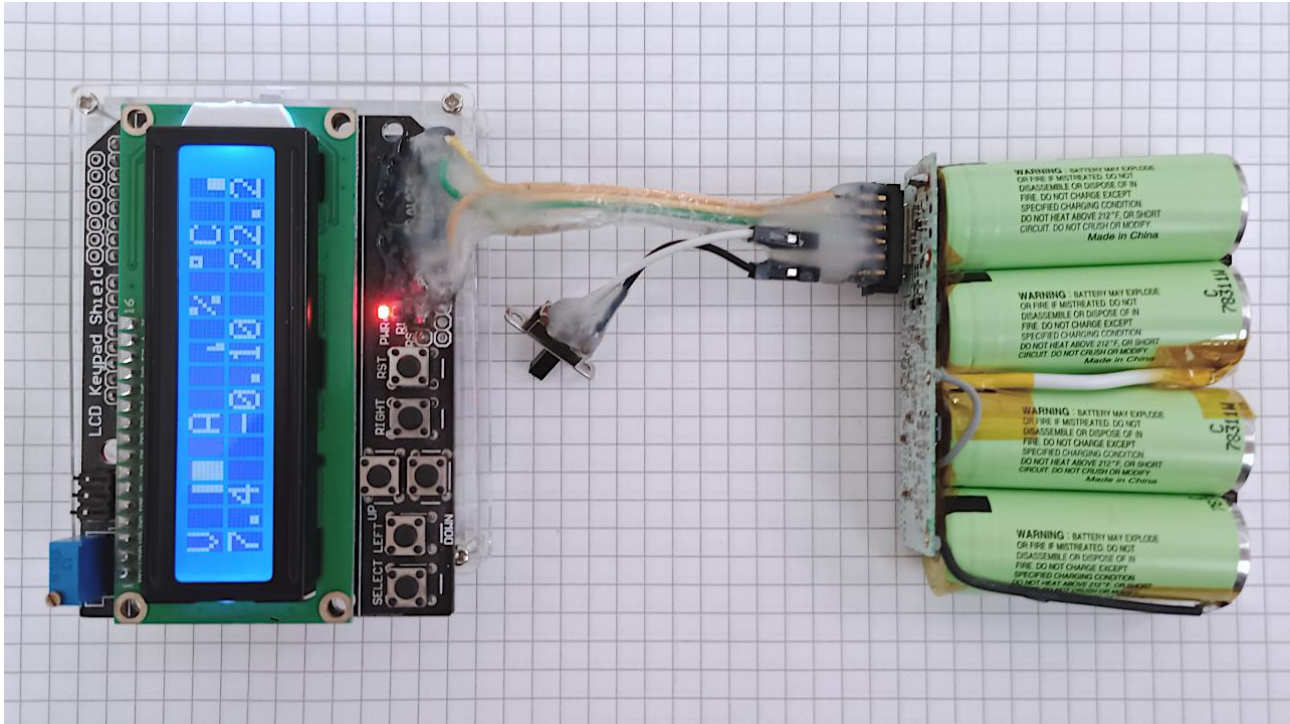


To avoid this pitfall, you need to temporarily solder a wire between test points TP4/TP5 and TP12. Only then you can solder the nickel plate (BAT+), the gray (BATM) and black (BAT-) wires to the PCB as initially. With your multimeter, verify the voltage at the terminals of the two sets of cells to respect the polarity. Feel free to reuse insulating tape in between.



Now, you should measure with your multimeter, that the resistance between TP4/TP5 (BAT-) and TP24 (GND/PACK-) is very high. If this is true, then you can use another wire to connect them for a very short time (a split second). IC1 reinitialization should be done.

Finally, measure again that the resistance between TP4/TP5 and TP24 is now very close to 0Ω , and unsolder the wire between TP4/TP5 and TP12. If you have the BattMon 2.0 toolkit, you can also check that the battery is working well and returning data such as Voltage, Current, State of charge, Temperature, Remaining capacity or Cycle count.



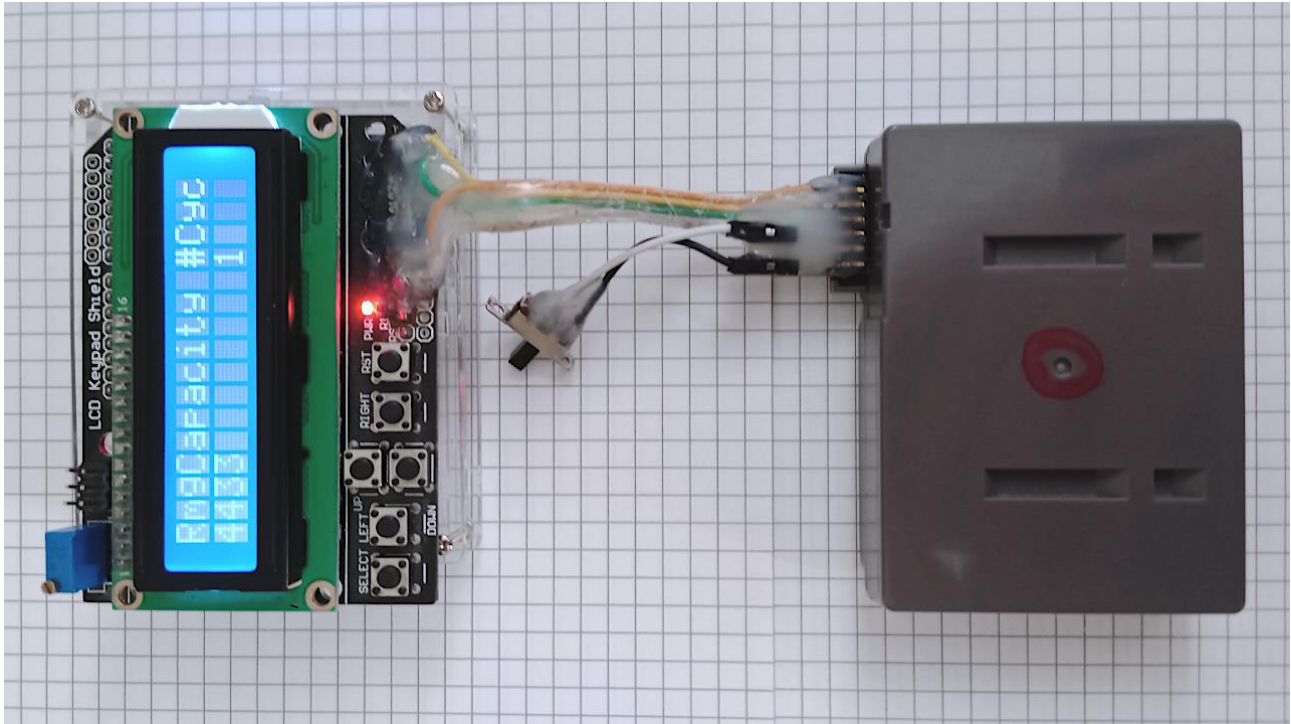
6. Reintegrate all in the case

For the final test, you have to put the pack back in its case, positioning the connector first.



Tips and tricks

To conclude, it will be useful to perform a full charge and then a full discharge, so that the bq2040 gas gauge IC41 determines the exact capacity of the battery. This calibration can be done either via the charger in discharge mode or with the BattMon 2.0 toolkit, when the voltage reaches a value between EDV1 (6.3V) and EDVF (6.0V), as indicated in the datasheet: "FCC is updated on the first charge after a qualified discharge to EDV1".



I accomplished this operation on four batteries. Two packs have been replaced with cells of around 1500mAh and two others with NCR18500A cells of around 2000mAh. I burned the fuse F1 several times, hopefully most of the components can still be found on internet.



Appendix A

All equipment is also easily found on the internet, here is the order price for some of them:

Spot welding machine + nickel roller: ~30\$/€/F from Aliexpress / ~50\$/€/F from Amazon

Kapton tape: ~3\$/€/F from Aliexpress / ~8\$/€/F from Amazon

HTRC B6 charger: ~25\$/€/F from Aliexpress / ~40\$/€/F from Amazon

Bus Pirate v3.6 + basic probe set: ~40\$/€/F from Aliexpress or Seeedstudio

<https://www.seeedstudio.com/Bus-Pirate-v3-6-universal-serial-interface-p-609.html>

Arduino Uno R3: ~10\$/€/F from Aliexpress / ~20\$/€/F from Arduino

<http://store.arduino.cc/products/arduino-uno-rev3>

LCD Keypad Shield: ~5\$/€/F from Aliexpress / ~10\$/€/F from Arduino

<https://www.dfrobot.com/product-51.html>

Appendix B

Here are the main datasheet and technical documents that should interest you to understand what is explained in this tutorial:

bq2040 gas gauge datasheet:

<https://www.ti.com/lit/ds/symlink/bq2040.pdf>

24LC01 EEPROM memory datasheet:

<https://ww1.microchip.com/downloads/en/devicedoc/21711c.pdf>

Bus Pirate specification:

http://dangerousprototypes.com/docs/Bus_Pirate

PuTTY client for windows:

<https://www.putty.org>

Arduino Uno R3 specification:

<https://docs.arduino.cc/hardware/uno-rev3>

Arduino Uno R3 schematic:

https://content.arduino.cc/assets/UNO-TH_Rev3e_sch.pdf

LCD Keypad Shield specification:

https://wiki.dfrobot.com/LCD_KeyPad_Shield_For_Arduino_SKU_DFR0009

LCD Keypad Shield schematic:

<https://image.dfrobot.com/image/data/DFR0009/LCDKeypad%20Shield%20V1.0%20SCH.pdf>

And here is the result of my research to make this battery work again:

ERA-201B1 wiring diagram with BattMon 2.0 toolkit:

https://github.com/lpollier/battmon/blob/master/example/ERA-201B1_wiring_diagram.png

ERA-201B1 reverse engineering schematic:

https://github.com/lpollier/battmon/blob/master/example/ERA-201B1_reverse_engineering_schematic.pdf

Identifying a SOIC-8 IC with "302F" marking:

<https://electronics.stackexchange.com/questions/550228/identifying-a-soic-8-ic-with-302f-marking>

The video of my first test made with a re-celled battery:

<https://youtu.be/zO9p-0TFVrM>

I want to say a special thanks to Chris, Andrej and Andrew for helping me improve this document.